Semestrálna práca 2

**Algoritmy a údajové štruktúry 2**

Autor: Bc. David Kučera

Štud.skup.: 5ZIS12

Obsah

[Popis semestrálnej práce 3](#_Toc184131511)

[Dátové štruktúry 3](#_Toc184131512)

[Zložitosti operácií – neutriedený súbor dát na disku 3](#_Toc184131513)

[Vyhľadanie 3](#_Toc184131514)

[Vloženie 3](#_Toc184131515)

[Vymazanie 4](#_Toc184131516)

[Zložitosti operácií – rozšíriteľné hešovanie 5](#_Toc184131517)

[Vyhľadanie 5](#_Toc184131518)

[Vloženie 5](#_Toc184131519)

[Vymazanie 5](#_Toc184131520)

[Architektúra práce 6](#_Toc184131521)

[UML diagram tried modulu FilesLib 6](#_Toc184131522)

[UML diagram tried modulu CarLib 7](#_Toc184131523)

[Grafické rozhranie aplikácie 7](#_Toc184131524)

[Vyhľadanie osoby 7](#_Toc184131525)

[Pridanie vozidla 8](#_Toc184131526)

[Pridanie návštevy servisu 8](#_Toc184131527)

# Popis semestrálnej práce

Cieľom semestrálnej práce bolo vytvoriť **demonštračnú verziu** softvéru pre informačný systém záznamov o návštevách autoservisu, pričom bolo požadované, aby tieto dáta boli uložené prevažne na pevnom disku, resp. pamäťovej karte, resp. vo vnútornej pamäti zariadenia.

V tejto verzii je umožnený výpis všetkých evidovaných údajov na obrazovku a nasledovné **operácie**:

1. **vyhľadanie** všetkých evidovaných údajov o vozidle,
2. **pridanie** **vozidla**,
3. **pridanie** **návštevy** **servisu**,
4. **zmena**,
5. **zmazanie** **návštevy** **servisu**,
6. **zmazanie vozidla**.

Najfrekventovanejšou operáciou, ktorá sa vykonáva je **vyhľadávanie**, a preto táto operácia pracuje s najväčšou možnou rýchlosťou.

# Dátové štruktúry

Ako dátové štruktúry pre uchovávanie údajov boli zvolené **rozšíriteľné hešovanie** a **neutriedený súbor dát na disku**. Zložitosti sú uvádzané značkou O(x, y), pričom x zodpovedá počtu čítaní súboru a y počtom zápisov do súboru.

Zložitosti operácií – neutriedený súbor dát na disku

### Vyhľadanie

Pri vyhľadávaní je potrebné daný blok načítať zo súboru a následne z dát nájsť hľadané. – **O(1, 0)**

### Vloženie

Pri vkladaní môže nastať niekoľko scenárov.

1. Ak **mám** nejaké voľné bloky na pridanie dát. Najskôr vyberám čiastočne plné, potom prázdne. Ak ani jedno nemám, tak vytváram nový blok. Najskôr si načítam daný blok, do ktorého sa budú vkladať dáta.
   1. Ak sa po vložení záznamu do bloku ešte **zmestia** ďalšie záznamy.
      1. Ak bol v zreťazení prázdnych blokov, tak je treba ho preradiť do zreťazenia čiastočne plných blokov.

* Nastavím nasledovníka ako začiatok zreťazenia.
  + Ak existuje, tak ho načítam, vymažem mu predchodcu a zapíšem do súboru. – **O(2, 2)**
* Blok zaradím ako začiatok zreťazenia čiastočne plných blokov
  + Ak začiatok zreťazenia už bol, načítam si ho, prepíšem mu predchodcu na vkladaný blok a zapíšem do súboru. – **O(2, 2)**
    1. Inak neriešim nič, lebo ak je už zaradený v zreťazení čiastočne plných blokov, tak tam aj ostane a nič sa nemení. – **O(1, 1)**
  1. Ak sa po vložení záznamu do bloku už ďalšie záznamy **nezmestia**.
     1. Ak má nejakého nasledovníka v zreťazení, je potrebné ho načítať, a vymazať mu predchodcu. Ak bol blok začiatkom zreťazenia, je treba tento začiatok previesť na nasledovníka. – **O(2, 2)**

1. Ak **nemám** žiadne voľné bloky – musím vytvoriť nový na konci súboru.
   1. Ak sa blok pridaním dát **zaplnil**, tak sa len zapíše na koniec súboru. – **O(0, 1)**
   2. Ak sa však do bloku ešte **zmestia** nejaké dáta, je treba ho zaradiť do zreťazenia blokov, nakoľko sme doň niečo vložili, budeme ho vkladať do zreťazenia čiastočne naplnených blokov. Je potrebné načítať prvý v danom zreťazení, tomu nastaviť ako predchodcu novo pridaný blok, novému bloku ako nasledovníka daný blok, tieto zmeny zapísať do súboru blokom a následne nastaviť novo vytvorený blok ako začiatok zreťazenia. Teda sme robili načítanie, zapísanie prvého bloku v zreťazení a potom zapísanie nového bloku. – **O(1, 2)**

Ak toto zreťazenie nemalo žiadny začiatok. - **O(0, 1)**

### Vymazanie

Pri mazaní sa vždy načíta blok na danej adrese a vyhodí sa z neho daný záznam. Na konci sa tento blok zapíše do súboru, aby sa zachovali zmeny.

Následne môže nastať niekoľko scenárov.

1. Ak blok **nie** **je** v žiadnom zreťazení. (bol plný)
   1. Ak blok obsahuje **nejaké** validné dáta – potrebné ho zaradiť do zreťazenia čiastočne plných blokov.
      1. Ak je adresa začiatkom zreťazenia, je to v poriadku, nič neriešim. – **O(1,1)**
      2. Ak je však adresa zreťazenia iná, musíme načítať blok začiatku zreťazenia, nastaviť mu ako predchodcu aktuálny blok a tieto zmeny zapísať. Ako začiatok zreťazenia nastaviť aktuálny blok. – **O(2,2)**
   2. Ak blok neobsahuje **žiadne** validné dáta, teda je prázdny – je potrebné ho zaradiť do zreťazenia prázdnych blokov.
      1. Ak už začiatok zreťazenia existuje, je potrebné si blok načítať, ako predchodcu mu dať aktuálny blok, aktuálnemu bloku nastaviť nasledovníka na tento blok a zmeny v ňom zapísať do súboru. Ako začiatok zreťazenia nastaviť náš blok. – **O(2,2)**
2. Ak blok **je** súčasťou zreťazenia. Súčasťou prázdnych blokov byť nemohol, nakoľko obsahoval nejaký záznam, ktorý sme mazali. Stačí teda odsledovať, či bol súčasťou zreťazenia čiastočne plných blokov a stal sa z neho prázdny blok.
   1. Ak sa z bloku stal vymazaním záznamu prázdny blok – treba ho vystrihnúť zo zreťazenia a zaradiť do zreťazenia prázdnych blokov.
      1. Ak mal nejakého nasledovníka, je potrebné ho načítať, nastaviť mu predchodcu na predchodcu aktuálneho bloku a zmeny zapísať do súboru. **+ O(1,1)**
      2. Ak mal nejakého predchodcu, je potrebné si ho načítať, nastaviť mu nasledovníka na nasledovníka aktuálneho bloku a zmeny zapísať do súboru. **+ O(1,1)**
      3. Vynulujeme zreťazenie bloku a zaradíme do zreťazenia prázdnych blokov.
      4. Ak začiatok zreťazenia už je, načítame si ho, ako predchodcu mu nastavíme aktuálny blok, aktuálnemu bloku nastavíme ako nasledovníka daný blok a zmenu zapíšeme do súboru. Nakoniec ako začiatok zreťazenia dáme aktuálny blok. **+** **O(1,1)**
   2. Inak neriešime nič. – **O(1,1)**

Na konci vymazávania sa ešte skontroluje, ak sme mazali z posledného bloku a celkom sa vyprázdnil, či sú na konci súboru prázdne bloky. Ak áno, súbor sa skráti o dané prázdne bloky.

**Mazanie od konca**

Postupne sa prechádzajú bloky od konca, kým sa nenarazí na blok, ktorý obsahuje nejaké validné dáta. Načíta sa posledný blok – **O(1,0)**. Ak neobsahuje žiadne validné dáta – vymaže sa, inak koniec. Ak je začiatkom zreťazenia prázdnych blokov, treba nastaviť začiatok na jeho nasledovníka.

1. Ak má blok nejakého nasledovníka – načítam ho, ako predchodcu mu nastavím predchodcu mazaného bloku a zapíšem do súboru. – **O(1,1)**
2. Ak má nejakého predchodcu – načítam si ho, ako nasledovníka mu nastavím nasledovníka mazaného bloku a zapíšem do súboru. – **O(1,1)**

Nakoniec súbor skrátim o daný blok.

## Zložitosti operácií – rozšíriteľné hešovanie

### Vyhľadanie

Vyhľadávanie pomocou štruktúry je veľmi dobré – **O(1,0)**.

### Vloženie

Vkladanie nových záznamov je pomocou štruktúry veľmi dobré, závisí od počtu prístupov neutriedeného súboru na disku, tie viď vyššie v dokumente.

Split blokov – nakoľko táto operácia „prerozdeľuje“ prvky medzi dva bloky, je potrebné oba načítať, v operačnej pamäti prepísať záznamy v daných blokoch a následne tieto upravené bloky zapísať do súboru. – **O(2,2)**

### Vymazanie

Štruktúra nepodporuje vymazávanie.

# Architektúra práce

Aplikácia je rozdelená do 3 hlavných programových modulov.

Prvým najnižším modulom sú **FilesLib**, kde sú implementácie dátové štruktúry, ktoré sa v aplikácií využívajú, teda súbor neutriedených dát na disku a rozšíriteľné hešovanie. *Na otestovanie týchto štruktúr je implementovaný aj tester operácií s CLI rozhraním v projekte FilesTest.*

Ďalším modulom je modul **CarLib**, ktorý obsahuje „jadro“ aplikácie. Využíva modul FilesLib a vykonáva nad ním operácie pre prácu s údajmi a štruktúrami. V rámci tohto modulu je implementovaný aj generátor dát a samotné štruktúry dát, ako napr. návštevy servisu, zákazníci.

Posledným najvyšším modulom je **CarViewer**, ktorý obsahuje GUI rozhranie pre prácu s modulom **CarLib**.



## UML diagram tried modulu FilesLib

Obrázok, na ktorom je text, snímka obrazovky, rad, diagram

Automaticky generovaný popis

V rámci tohoto modulu sú implementované dátové štruktúry súboru neutriedených dát na disku a rozšíriteľné hešovanie. K týmto štruktúram prislúchajú bloky **Block** a **ExtendibleHashFileBlock**, ktoré uchovávajú dáta. Taktiež sú v tomto module definované rozhrania, ktoré sa využívajú. Blocky implementujú rozhranie **IRecord**, pretože sa ukladajú do binárneho súboru. Následne prvky T vkladané do **HeapFile** musia implementovať rozhranie **IData**. Nakoľko ExtendibleHashFile pracuje so štruktúrou **HeapFile**, je nutné, aby prvky T vkladané do tejto štruktúry implementovali rozhranie **IHashable**, ktoré musí byť potomkom rozhrania **IData**.

## UML diagram tried modulu CarLib

Obrázok, na ktorom je text, snímka obrazovky, písmo, rad

Automaticky generovaný popis

V tomto module je implementované „jadro“ aplikácie. Je v ňom implementovaný generátor dát **DataGenerator**. Taktiež sú v tomto module aj štruktúry dát – **Person**, **Visit**, **PersonId**, **PersonEcv**. PersonId a PersonEcv sa ukladajú len do rozšíriteľného hešovania. Trieda **CarSys** obsahuje všetky potrebné metódy, ktoré sú potrebné pre správne fungovanie GUI.

# Grafické rozhranie aplikácie

Aplikácia má jednoduché používateľské rozhranie.

## Vyhľadanie osoby

Vyhľadať osoba sa dá buď pomocou ID alebo EČV. Zadáme dané údaje do blokov a následne stlačíme tlačidlo „Nájdi podľa ID“ alebo „Nájdi podľa „EČV“. Po vyhľadaní nejakej osoby zobrazí všetky uložené dáta o nej, vrátane všetkých návštev. Tie je však nutné rozkliknúť, pričom sa po kliku zobrazia v spodnom bloku „Notes“, pričom jeden riadok zodpovedá jednej poznámke.

Obrázok, na ktorom je text, softvér, číslo, snímka obrazovky

Automaticky generovaný popis Obrázok, na ktorom je text, softvér, počítačová ikona, webová stránka

Automaticky generovaný popis

Zobrazenie všetkých evidovaných dát – obrázok vľavo bez rozkliknutia návštevy, pravý s rozkliknutou návštevou servisu

## Pridanie vozidla

Nové vozidlo pridáme pomocou Menu – Edit – Pridaj záznam o vozidle. Následne sa nám zobrazí okno, ktoré nás vyzve pre zadanie ID osoby. Ak zadané ID už v databáze existuje, vyskočí chyba. Ak je zadané ID unikátne, následne sa zobrazí okno, ktoré nás vyzve pre zadanie EČV vozidla. Ak zadané EČV už v databáze existuje, vyskočí chyba. Ak je ID v poriadku, zobrazí sa nám nový záznam na hlavnej obrazovke. Následne môžeme pomocou ID alebo EČV osobu dohľadať. Ak jej chceme zmeniť nejaké atribúty, môžeme ich vyplniť v hlavnom okne podľa potreby. Po editácií je však nutné zmeny uložiť pomocou tlačidla „Uprav“.

Obrázok, na ktorom je text, softvér, počítačová ikona, webová stránka

Automaticky generovaný popis

Tlačidlo v menu, ktoré pridáva nový záznam do databázy

## Pridanie návštevy servisu

Pre pridanie novej návštevy servisu musíme vyhľadať danú osobu a následne kliknúť na tlačidlo „Pridaj návštevu...“. Následne sa nám otvorí okno, kde môžeme zadať informácie o danej návšteve servisu, ako dátum, cenu a poznámky. Akonáhle tieto dáta vyplníme a stlačíme tlačidlo „OK“, sa nám nová návšteva zobrazí medzi návštevami zákazníka. Je nutné však pre uchovanie týchto dát stlačiť tlačidlo „Uprav“, ktoré dáta zapíše na disk. Ak sa toto tlačidlo nestlačí, dáta sa neprepíšu a ostanú pôvodné. To platí pre zmenu všetkých atribútov, nie len návštev.

Obrázok, na ktorom je text, číslo, softvér, snímka obrazovky

Automaticky generovaný popis

Ak máme návštevu servisu rozkliknutú, je možné ju aj zmazať pomocou tlačidla „Odober návštevu“. Po tejto úprave nám zmizne z bloku návštev. Je však ale potrebné, rovnako ako vo všetkých úpravách, danú úpravu potvrdiť tlačidlom „Uprav“.